SUMMARY
In this project you will create a new class for picture albums, using the Picture class. It will continue your practice of keeping a VERSION HISTORY which was taught and required in the course.
Just as a music album contains many music tracks, and a stamp album contains a stamp collection, you will make a class to contain a collection of Pictures. Such classes are called container classes.
YOUR SUBMISSION MUST BE A VERSION HISTORY, with separate versions in which one of the two constructors was started and later completed first! Upload to Blackboard a .zip file containing ALL THE VERSION. By this time in the course, submissions that are NOT version histories will get 0 points!

DUE TIME
ON-TIME: December, May 8, 11:59PM. (-7% per day late after that, +(7%) per day before that!)

PREPARATION
Chapter 11 of the text, but instead of a Student with multiple constructors and instance methods you will make an Album which collects Pictures instead of grades.

DESCRIPTION
You must create one class named Album. So you must start editing a file named Album.java to contain the class definition:

```java
public class Album {  }
```

The Album class might have these fields:

1. (Mandatory!) A Picture array reference named pictArray
   It will be initialized by `pictArray = new Picture[capacity];` As Pictures are added one-by-one, references to them will be stored in this array so that they are available for copying LATER, at some time AFTER the correct values in width and height (see above) have been computed.

2. (Mandatory!) An int named nPictsInAlbum. Its value will start at 0 and be used to track the number of Picture s that have been added so far. It will be used to tell where in the array below another Picture reference can be put.

3. (Optional. You can use the length of the array instead.) An int named capacity
   Its value will be set by the constructor (see below) that enables us to specify the maximum number of Pictures that may be added to the Album.

4. (Optional) An int named width
   Its value is the width of the Picture that would be made from the Pictures currently stored in this Album. When a Picture is added, this value should be changed. (Or the width might be computed only when the the combination Picture about to be made.)

5. (Optional) An int named height
   Analogous to width. When a Picture is added, this value might or might not have to be changed. (Ask yourself why?)

6. (Optional) A Picture named result
   Its value will be the reference to the Picture object into which the colors of the Pixels of the Pictures in the Album will be copied.

7. (Optional) An int named nextX
   Its value will start at 0 and be used to track the X-coordinate at which to begin copying the next Picture. Or, it might be local to the method that controls the copying of the Picture s one by one.
The Album class will have 1 constructor:

(Remember that a constructor is a special kind of method that is called automatically just after and only when the new operation is done to make a new Album.)

    public Album( int capacityParam )

This constructor enables the program using Album to set up an Album that will have the accept up to the number of Picture s given by the value of capacityParam.

The Album class will have 3 public methods: Two have the same name but different parameters! (The technical term for such different methods is "polymorphic") They are really different methods.

    public boolean addPicture(Picture thePicture)

This method tries to add the given Picture to the Album, so when the Album is shown (in the future!), the given Picture appears at the (right) end. It returns true if the Picture is successfully added and false otherwise.

    public void explore()

A new Picture is made whose height is the max of the heights, and width is the sum of the widths of the stored Pictures. The Pictures in the Album so far are copied into a new Picture and that Picture is displayed by calling the explore() method on it. All the added Picture copies should start at the top of the new Picture and be laid out horizontally, with no overlaps or gaps.

    public boolean addPicture(Picture thePicture, int where)

Instead of adding the given Picture at the end, this method inserts it at the position given by the where parameter value. That value should be between 0 and nPictsInAlbum.

When it is 0, the given Picture is inserted at the beginning and the other Pictures are shifted right to make the space for it. When it is nPictsInAlbum, it just adds the given Picture at the right end. (When in-between, it adds the given Picture in-between the where-th Picture and the Picture after that one.)

Another way to explain it, from Jeremy: where is a value ranging from 0 to the number of Pictures in the Album, inclusively. The new Picture should be inserted at the index where (remembering that the indices start at 0), and all old the Picture (refs) from index where to the end should be shifted to the right.

DON'T BEGIN PROGRAMMING THIS until you ACTUALLY DO sketch examples on paper, as we showed in the lecture!!! It really is tricky for beginners.

And you might play with making a row of coins, etc. like I did with the chairs, to get the feel of how array elements must be moved around to make room to insert a new value into one of the elements, and not lose the old values!

This method should return true if there is capacity in the array AND the value of where locates a valid location to insert another Picture. Otherwise, it must return false, and do no harm.

TESTING AND CREDIT
Submit Album.java in the LAST VERSION DIRECTORY that (1) must compile and (2) be testable with our AlbumTester.java program that will be available on the website. Otherwise, 0 points automatically! In other words: TEST IT WITH OUR TESTER...REALLY! Partial credit: The addPicture(Picture thePicture, int where) polymorph counts for the last 20 of 100 points. (revised 12/5/2012)